World Wide Web
https://doi.org/10.1007/s11280-021-00875-6

Temporal high-order proximity aware behavior analysis on Ethereum

Xiang Ao[1,2] · **Yang Liu**[1,2] · **Zidi Qin**[1,2] · **Yi Sun**[2,3] · **Qing He**[1,2]

Received: 20 May 2020 / Revised: 19 February 2021 / Accepted: 1 March 2021 /
Published online: 25 March 2021

Abstract

Ethereum, the most popular public blockchain with the capability of smart contracts and the cryptocurrency Ether, is escalating in the number of account addresses and transactions since its birth. Due to the decentralisation of the Ethereum blockchain and the anonymity of its users, Ethereum serves as a noteworthy environment for malicious activities that are difficult to unearth. As a result, understanding the behaviors of the account addresses on Ethereum has become an imperative problem receiving much attention very recently. Existing works for such task mainly rely on extracting statistical features of account addresses and applying machine learning techniques to group or identify them. However, seldom prevailing approaches take temporal information and high-order interactions among the account addresses into consideration. To this end, we propose a novel approach coined THCD (**T**emporal **H**igh-order proximity aware **C**ommunity **D**etection) for behavior analysis on Ethereum from the perspective of graph mining. First, frequent temporal motifs are mined over a transaction graph constructed by the Ethereum block transactions. Next, we define the high-order proximity between two accounts based on these temporal motif occurrences. Finally, a novel temporal motif-aware community detection method is devised to find account communities over the defined high-order proximity. Experiments on four real datasets constructed from Ethereum blocks demonstrate the effectiveness of our approach. Some discovered suspicious accounts are confirmed by real-world reports. Meanwhile, THCD is scalable to large-scale transaction datasets.

Keywords Temporal motif · Community detection · Ethereum · Blockchain

This article belongs to the Topical Collection: *Special Issue on Emerging Blockchain Applications and Technology*
Guest Editors: Rui Zhang, C. Mohan, and Ermyas Abebe

✉ Xiang Ao
aoxiang@ict.ac.cn

Extended author information available on the last page of the article.

1 Introduction

Ethereum, proposed in late 2013[1] and first launched in 2015, is one of the largest and most widely used blockchain platforms in the world. By the time of December 2019, there have been more than 80 million distinct addresses on it with processing a total of over 600 million transactions. The number of addresses escalates more than 20 times up from around 4 million addresses on July 3, 2017.

Anonymity and decentralisation are generally two major characteristics of Ethereum. First, the Ethereum platform does not store any personal information about users to protect their privacy, and it is known as the anonymity. However, such anonymity also might help miscreants who take part in money stealing or laundering to evade regulation since transactions can be completed without intermediaries and no one knows who conduct these transactions. For example, according to the research by Carbon Black[2], a total of $1.1 billion in cryptocurrency-related thefts happened in the first half of 2018, and money laundering was staged with the highest proportion on mixers[3] and online gambling sites. Second, every transaction is fully public and transparent to every account on Ethereum, which enables individual account to synchronize and participate in authenticating all transactions on the platform. It is known as the decentralisation of blockchain, which provides sufficient data for transaction analysis to understand and mitigate the menace of malicious behaviors on the Ethereum platform.

As a result, several recent efforts [4, 5, 14, 21, 23, 24] have begun to analyze transactions on Ethereum to understand the behaviors of the accounts within the platform. For example, [14, 21, 23] extracted statistical features for each account and adopted k-means clustering algorithm to find outliers that could be anomalous accounts. [5] characterized three major activities on Ethereum, namely money transfer, smart contract creation and smart contract invocation, as graphs and applied statistical cross-graph analysis to address attack forensics and anomaly detection tasks on Ethereum.

Despite the initial success achieved by the existing research, they may suffer from the following drawbacks on understanding and modeling complex behavior patterns of the accounts on Ethereum.

First, seldom prevailing methods take temporal information into consideration. However, the rich temporal information associated with transactions is crucial for understanding the accounts' behaviors. For instance, the fabricated transactions created by the exchanges making exaggerated trading volume are required to be completed as soon as possible to avoid unexpected price fluctuations. As a consequence, there might be a serendipitous proliferation in a number of transactions in some small-time periods, but no obvious abnormality in long-term statistics. It motivates us to leverage temporal information to describe accounts' behaviors at a finer granularity.

Second, high-order interactions among accounts and addresses fail to be fully exploited. Major current researches solely consider each account itself or the direct relation between the sender and the receipt of the transaction. However, paradigms of the platform and

[1] https://github.com/ethereum/wiki/wiki/White-Paper

[2] https://www.carbonblack.com/2018/06/07/carbon-black-threat-report-cryptocurrency-gold-rush-dark-web/

[3] A mixer is an online software service that can swap cryptocurrencies for ones with different transaction histories.

some specific purposes may derive high-order interactions among accounts or addresses, which might become essential features to understand account behaviors on Ethereum. For example, when transferring funds into an exchange, the deposit address will be used as an intermediary due to the paradigms of Ethereum. Such deposit address correlates both users and the hot wallet of the exchange. Similarly, the withdrawal address is performed as the intermediary if a person wants to get his/her funds out of exchanges. For miscreants who launder money, as another example, they might manipulate a group of accounts to accomplish crime-specific tasks, which might incur such group of accounts possess correlated and analogous behaviors but never form a transaction with each other. Hence, taking high-order interactions among accounts into consideration to capture more substantial features for identifying behavior patterns on Ethereum is in urgent demand.

To this end, we propose **T**emporal **H**igh-order proximity aware **C**ommunity **D**etection (THCD for short) for behavior analysis on Ethereum. First, a transaction graph describing fund flows among accounts is constructed from the Ethereum transactions, and an efficient frequent temporal motif mining method is adopted to obtain the frequent transaction patterns (temporal motif) in the graph. Next, we define the high-order proximity between two accounts as the co-occurrence count in transaction pattern occurrences, and we can derive a high-order proximity matrix of accounts based on these frequent transaction patterns. Finally, a novel temporal motif-aware community detection method is devised to find account communities over the defined high-order proximity matrix. We conduct extensive experiments on four real datasets from Ethereum. The results demonstrate that our method can not only outperform SOTA community detection methods in quantified measures but also unearth some suspicious account communities that are confirmed by real-world reports.

The main contributions of our work are the following three-fold.

- We re-examine the behavior analysis of Ethereum by incorporating both temporal information and high-order interactions among accounts, which is seldom considered by previous studies.
- We propose a novel temporal high-order aware community detection method, named THCD, for behavior analysis on Ethereum. THCD centers on three technical components, namely an efficient temporal-motif mining algorithm, a high-order proximity computation approach and a temporal motif-aware community detection method.
- Experimental results on the real datasets from Ethereum illustrate that THCD is able to discover abnormal accounts underpinned by high-order proximities. Some detected fake volume exchanges and suspicious money laundering accounts are confirmed by real-world reports.

The remainder of this paper is organized as follows. We summarize the related work in Section 2. Section 3 presents the concepts and the research problem of this paper. Section 4 details our proposed model THCD. Section 5 explains experimental steps including data collection and parsing, evaluation methods, and experimental results. We finally finish the paper with a conclusion in Section 6.

2 Related work

The existing studies related to our work are categorized as follows.

2.1 Behavior analysis in Ethereum

Ethereum is an account-based blockchain implementation which supports smart contracts to be deployed on it. There are generally two kinds of accounts on Ethereum, namely external owned account (EOA) and smart contract account. An EOA is created by individual user in the external world, and a smart contract account corresponds to a deployed contract. Various behaviors of accounts on Ethereum, including fund transfers, contract creation and contract invocation, are recorded in transactions, which sheds new light on understanding and uncovering illegal activities over the network such as money laundering, bribery, phishing, fraud, among others.

Hence, analyzing transactions on Ethereum to understand behaviors of accounts emerges in computer science research communities. One of the prevailing technical route is utilizing machine learning methods for behavior analysis on Ethereum. To name some, [8] conducts an XGBoost classifier on the transaction histories of accounts to detect the illicit ones. [28] presents a behavior-aware profiling of individual smart contract on Ethereum. [6, 7] build classification models to detect latent Ponzi schemes implemented as smart contracts.

Another major technical route is to leverage graph analysis techniques. For example, [5] leverages cross-graph analysis to characterize three major activities on Ethereum to detect security issues. [25] proposes a node clustering approach for user identity discrimination and malicious user detection. [18] designs several flexible temporal walk strategies for random-walk based graph representation of the transaction network and benefits for the downstream link prediction task.

However, they seldom incorporate both temporal information and high-order behavioral interactions into their approaches. In this work, the proposed THCD discovers frequent temporal motifs to capture the temporal interactions among multiple accounts, and the devised temporal-motif aware community detection algorithm underpins high-order proximity preserved behavior analysis. It is orthogonal to previous work.

2.2 Behavior analysis in cryptocurrency

Apart from Ethereum, we also observe related researches for behavior analysis in other blockchain based cryptocurrencies. For instance, pump-and-dump schemes [17] are fraudulent price manipulations through the spread of misinformation. A case study in [29] investigates 412 pump-and-dump activities organized in Telegram channels and discovers patterns in crypto-markets associated with pump-and-dump schemes. Anomaly detection techniques are utilized in [13] to locate points of anomalous trading activity in cryptocurrencies. [23] utilizes the One Class Support Vector Machines algorithm to detect outliers in cryptocurrency transactions, and the k-means algorithm is applied to group the similar outliers with the same type of anomalies. The k-means algorithm is also used in [14] for monitoring and clustering malicious activities in the node behaviors by separating groups with similar traits from blockchain networks. Graph Convolutional Networks are exploited in [27] for anti-money laundering in Bitcoin.

Most of the above approaches rely on labeled accounts to train the model, however, it might be inefficient and biased in anonymous blockchain settings due to the expensive acquisition cost and the incomplete labeled accounts. While in this work, our THCD adopts temporal motifs mining techniques on the graph constructed by massive transactions, which incorporates the temporal high-order proximity of accounts in an unsupervised manner, which is distinct to existing studies.

2.3 Graph mining and analysis

Our work is also related to graph mining and analysis. There are fruitful efforts about this topic, and we summarize graph pattern mining [1, 10] and community detection [3, 16] here, considering they are closer to our work.

Graph pattern, also known as motif or graphlet, refers to interconnections occurring in graphs at numbers that are significantly higher than those in random graphs [19]. The task of graph pattern mining [1, 10] is to find these frequent sub-graphs and their occurrence counts. These sub-graphs help to understand the higher-order organizations [2] in graph since we can define higher-order proximity based on them. The high-order proximity in this paper is different from them because we simultaneously consider the temporal information, which constrains temporal orders of the edges in the sub-graphs.

Community detection aims to partition the whole graph into some sub-parts [3, 9, 11, 12, 16, 20, 26]. Modularity is most commonly used measure of the quality of the partition. Louvain [3] is a representative greedy algorithm to optimize modularity. There are also other optimization strategies like simulated annealing [9], spectral optimization [20], and so on. Recently, motif-aware community detection methods [16, 26] have been developed to preserve the high-order structure of the graph. However, none of them takes the temporal motif into consideration. Recently, researchers also extend the community detection methods to dynamic graphs [11, 12] in which the graphs might evolve as time changes.

In this work, we combine temporal motif mining and a specialized temporal high-order proximity preserved community detection approach to understand the behaviors of accounts on Ethereum, which is a brand new application of graph analysis techniques on blockchain transaction monitoring.

3 Concepts and overview

In this section, we first introduce the concepts which will be used in our paper. Then we give an overview of the proposed framework of THCD.

3.1 Definitions and concepts

Definition 1 (Transaction) A transaction on Ethereum is generally denoted by a 3-tuple $\tau = (u, v, t) \in \mathcal{T}$, where u and v are the sender and recipient address (also known as account), respectively, t is the timestamp that this transaction has been included in a newly mined block. $\mathcal{T}$ refers to the transaction set that stores all transactions on Ethereum.

Besides the information mentioned in Definition 1, a transaction on Ethereum may contain other fields like data, transfer amount, input, gas price and gas limit. The data field is mainly for contract related activities and always left empty in fund transfer. On the contrary, the transfer amount field can be greater than zero only in a fund transfer transaction. The gas price and limit fields are related to the cost processing the transaction, which is out of the scope of this paper. Therefore, we omit these fields in Definition 1 for convenience.

Definition 2 (Transaction Graph) An Ethereum transaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a temporal graph which is built upon the transaction set, i.e. $\mathcal{T}$, where $\mathcal{V}$ is the account set and $\mathcal{E}$ is the collection of temporal edges. Each temporal edge $e_i \in \mathcal{E}$ is represented by 3-tuple

$(u_i, v_i, t_i), i = 1, \ldots, |\mathcal{E}|$, such that (u_i, v_i, t_i) is a transaction, i.e., $(u_i, v_i, t_i) \in \mathcal{T}, u_i, v_i \in \mathcal{V}$ and $t_i \in \mathbb{R}$ is a timestamp.

A toy example of transaction graph is shown in Figure 1. The node is represented by $a - f$ and the numbers attached to the edge denote the timestamps. For instance, 3, 7, 8 indicates transaction $a \to b$ occurs at $t = 3, 7, 8$, respectively. Therefore, there are totally three temporal edges between node a and b. For the sake of simplicity, we only draw single arrow to indicate temporal edges with the same start and end nodes and differentiate them with the attached timestamps.

Definition 3 (Temporal Motif) A temporal motif is a totally ordered sequence of temporal edges, denoted as $M = \langle (u_1, v_1, \underline{1}), (u_2, v_2, \underline{2}) \ldots, (u_\ell, v_\ell, \underline{\ell}) \rangle$ such that $u_i, v_i \in \mathcal{V}$. The nodes in a temporal motif M could be same though they are denoted by different symbols. The numbers with the underline $\underline{1} \sim \underline{\ell}$ indicate the time order. A temporal motif that contains k distinct nodes, ℓ temporal edge is called a k-node, ℓ-edge temporal motif.

For example, Figure 2a demonstrates an example of a 3-node, 3-edge temporal motif, which could be represented by $M = \langle (u, v, \underline{1}), (w, v, \underline{2}), (w, v, \underline{3}) \rangle$. It means after u transferred fund to v, w transferred to v consecutively for two times. The numbers with underline $\underline{1}, \underline{2}, \underline{3}$ indicate the relative time order and all the transactions.

Definition 4 (Temporal Motif Occurrence) An occurrence of a k-node, ℓ-edge temporal motif could be a subgraph of the transaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Such an occurrence is denoted by $\mathcal{H} = (\mathcal{V}_h, \mathcal{E}_h)$, where $\mathcal{V}_h \subset \mathcal{V}$, $\mathcal{E}_h \subset \mathcal{E}$, $|\mathcal{V}_h| = k$ and $|\mathcal{E}_h| = \ell$. Sorting all the temporal edges in $\mathcal{E}_h$, we have an ordered sequence like $\langle (u_1, v_1, t_1), (u_2, v_2, t_2) \ldots, (u_\ell, v_\ell, t_\ell) \rangle$ such that $t_1 < t_2 \ldots < t_\ell$, $t_\ell - t_1 \leq \delta$, where $u_i \in \mathcal{V}_h$, $v_i \in \mathcal{V}_h$, $i = 1, \ldots, \ell$, and δ is a user-specified threshold called maximum time interval.

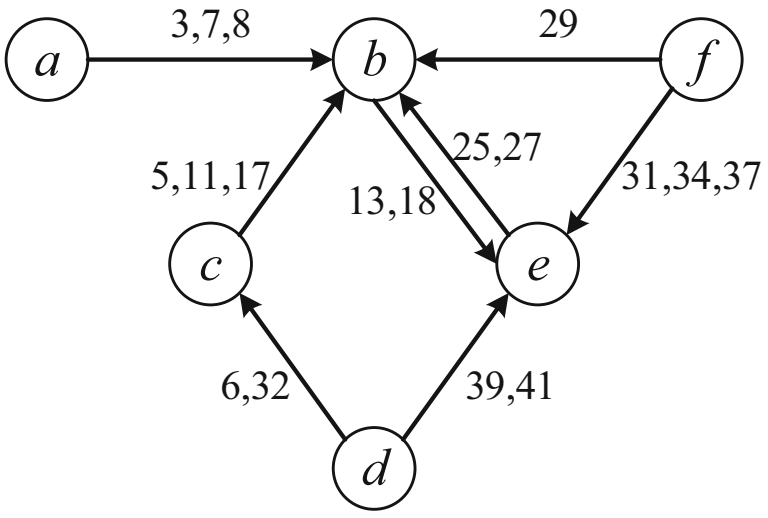

Figure 1 A toy example of transaction graph. Each lowercase letter represents a node and each number denotes a timestamp of the corresponding temporal edge

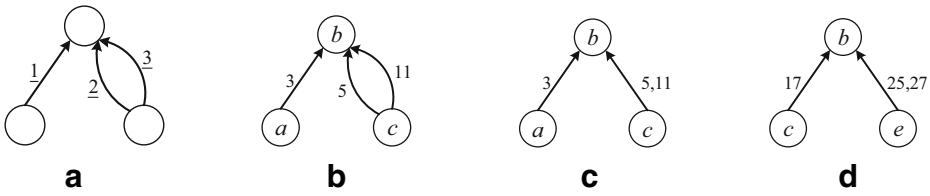

Figure 2 An example temporal motif and its occurrences. The numbers with underline in (**a**) indicate the time order while in (**b**)-(**d**) are the timestamps

For example, Figure 2b and d demonstrates two occurrences of the example 3-node, 3-edge temporal motif shown in Figure 2a from the transaction graph in Figure 1 if we set $\delta = 10$, i.e., $\langle (a, b, 3), (c, b, 5), (c, b, 11)\rangle$ and $\langle (c, b, 17), (e, b, 25), (e, b, 27)\rangle$, respectively.

Definition 5 (Frequent Temporal Motif) The support of a temporal motif M in a transaction graph $\mathcal{G}$ is defined as the number of occurrences of M in $\mathcal{G}$, denoted by $\text{sup}_{\mathcal{G}}(M)$. If the support of a temporal motif exceeds the a user-specified threshold *min_sup*, i.e. $\text{sup}_{\mathcal{G}}(M) \geq$ *min_sup*, we call such temporal motif as a frequent temporal motif.

Frequent temporal motifs in the transaction graph are also called transaction patterns in this paper, and we will use these two terms interchangeably hereafter. It is not trivial to define a proper threshold for frequent temporal motifs on the transaction graph of Ethereum since we have few prior knowledge on it. As a result, in this paper, we rank the support of all the k-node, ℓ-edge temporal motifs in the transaction graph given a maximum time interval threshold δ and select the top-K transaction patterns to be frequent. We will show in the experiment that these top-K transaction patterns are effective in understanding the behavior of accounts on Ethereum.

Definition 6 (Temporal High-order Proximity) For a given temporal motif M, the set of all its occurrences on a transaction graph $\mathcal{G}$ is denoted as $\text{occSet}_{\mathcal{G}}(\text{M})$. $\mathcal{V}_M$ is the account set contained in $\text{occSet}_{\mathcal{G}}(\text{M})$. For any $u \in \mathcal{V}_M$ and $v \in \mathcal{V}_M$, their temporal high-order proximity, denoted by $\sigma(\text{u}, \text{v})$, is defined to be the number of co-occurrences (u and v appear in the same temporal motif occurrence) in $\text{occSet}_{\mathcal{G}}(\text{M})$. We denote $\mathcal{W}_M$ as the temporal high-order proximity matrix among accounts underpinned by M, which is a symmetric square matrix having $|\mathcal{V}_M|$ rows and columns and $\mathcal{W}_M[u, v] = \mathcal{W}_M[v, u] = \sigma(\text{u}, \text{v})$.

For example, given the transaction graph $\mathcal{G}$ shown in Figure 1 and consider the temporal motif M shown in Figure 2a as a transaction pattern, then $\sigma(b, e) = 1$ since these two accounts are occurred in one temporal motif occurrence, namely $\langle (c, b, 17), (e, b, 25), (e, b, 27)\rangle$.

Definition 7 ($\mathcal{W}_M$-based Account Community) Given a temporal high-order proximity matrix among accounts $\mathcal{W}_M$, a $\mathcal{W}_M$-based account community is defined as a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_C\}$ of $\mathcal{V}_M$, where $\mathcal{V}_M \subset \mathcal{V}$ (c.f. Definition 6) and $\mathcal{C}_i \subset \mathcal{V}_M$, $i = 1, \ldots, C$, such that each account in $\mathcal{V}_M$ is included in a specific $\mathcal{C}_i$ for $1 \leq i \leq C$.

With all the concepts introduced, we describe the research problem of this paper. Given a transaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ on Ethereum and user specified parameters k, ℓ and δ, our problem is to discover transaction patterns M on $\mathcal{G}$, and detect $\mathcal{W}_M$-based account communities to understand the behaviors of Ethereum accounts.

3.2 Framework

In this subsection, we present the whole framework of THCD. The processing pipeline is exhibited as Figure 3.

Firstly, we run an Ethereum client to synchronize all the transaction blocks (c.f. Figure 3a) from the Ethereum network. Next, a transaction graph (c.f. Figure 3b) is constructed based on these transaction blocks. Then we adopt an efficient algorithm to calculate the support of all the k-node, ℓ-edge, δ-time temporal motifs to figure out which motifs are frequent in the transaction graph, and these frequent temporal motifs are also known as transaction patterns (c.f. Figure 3c). Next, we devise an algorithm to calculate the high-order proximity among accounts following its definition, and the temporal high-order proximity matrix of accounts is derived (c.f. Figure 3d). Finally, we propose a novel community detection method on such temporal high-order proximity matrix and find account communities underpinned by temporal high-order relationships among accounts for further analysis (c.f. Figure 3e). In the following, we will center on the three technical parts, namely transaction pattern mining algorithm, temporal high-order proximity calculation method and the temporal high-order community detection approach, respectively, to detail our THCD approach.

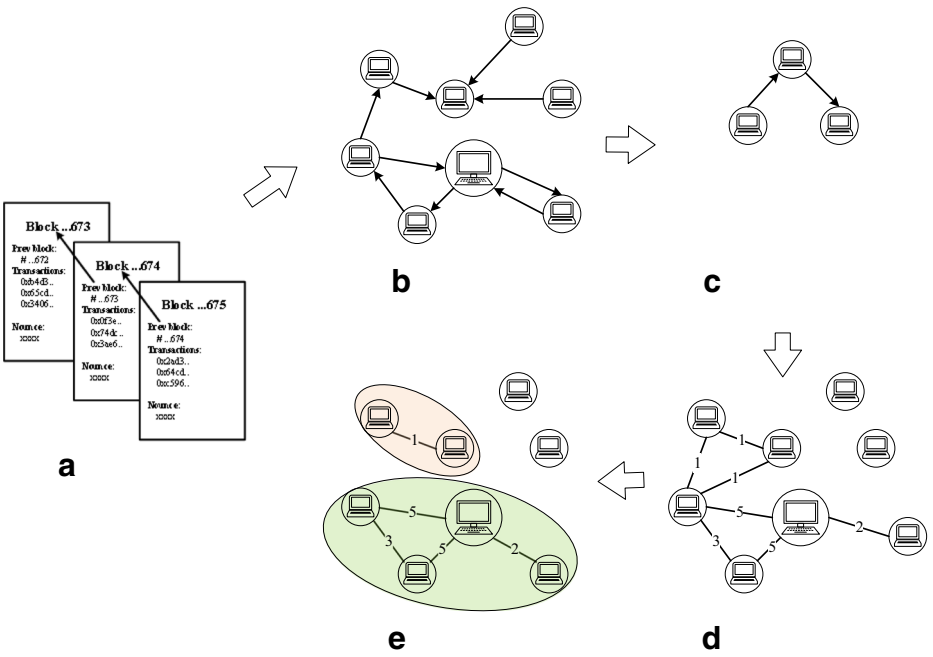

Figure 3 The pipeline of the proposed THCD

4 The THCD approach

This section details the backbone technical components of the proposed THCD approach, namely transaction pattern mining, temporal high-order proximity calculation, and temporal high-order aware community detection.

4.1 Transaction pattern mining

The goal of transaction pattern mining is to figure out which kinds of temporal motifs are frequent in the transaction graph. Here temporal motifs consisting of three nodes and three edges are specially considered since the high-order proximity usually captures the interconnections among more than two nodes and multiple edges, and the 3-node, 3-edge, δ-time temporal motifs are the most essential and simple instances. For the mining algorithm, we directly employ a fast counting algorithm proposed by Paranjape et al. [22] to accomplish this task. In [22], the proposed algorithm leverages dynamic programming approaches and heuristics to optimize the efficiency in counting the support of given 3-node, 3-edge, δ-time temporal motif instances. We omit more details of this algorithm as it is not the most important technical contribution of this paper, and the details of the mining algorithm can be referred in [22].

4.2 Temporal high-order proximity calculation

In this subsection, we explain how to compute the temporal high-order proximity and build the temporal high-order proximity matrix in Definition 6. Recall that transaction patterns that consists of three nodes and three edges can be discovered by the algorithm in [22]. Then for a given transaction pattern, the proximity between two accounts is defined to be their co-occurrence counts in this transaction pattern. Therefore, we devise an algorithm to count the co-occurrence number of any node pair in the transaction graph, which is summarized in Algorithm 1.

Given a transaction graph and the 3-node, 3-edge, δ-time temporal motifs, the algorithm is iterated over all the nodes of the transaction graph. For each node, we firstly gather all the neighbors (Line 3) and enumerate all the neighbor pairs (Line 4). Taking node b in Figure 1 as an example, the neighbor set of node b is $\mathcal{N}_b = \{a, c, e, f\}$ and the node pairs could be enumerated as $\{(a, c), (a, e), (a, f), (c, e), (c, f), (e, f)\}$.

For each pair of neighbors, we gather all the temporal edges between the center node and the neighbor node. Then these edges are sorted according to their timestamps (Line 6). For example, node a and node c are a neighbor pair of node b. The ordered temporal edge sequence is $S_b = \langle e_1 = (a, b, 3), e_2 = (c, b, 5), e_3 = (a, b, 7), e_4 = (a, b, 8), e_5 = (c, b, 11), e_6 = (c, b, 17)\rangle$.

While the algorithm is able to count the occurrences of multiple motifs simultaneously, we only exhibit the edges related to the example motif $M = \langle(\beta, \alpha, \underline{1}), (\gamma, \alpha, \underline{2}), (\gamma, \alpha, \underline{3})\rangle$, and the execution steps are presented in Table 1. We set $\delta = 10$ in this example.

The execution steps in Table 1 are demonstrated as follows. Since the length of sequence S_b is 6, the variable end loops from 1 to 6 (Line 14). For $t_{\text{start}} = 3$ and $t_{\text{end}} = 5, 7, 8, 11$, the while condition (Line 15) is not satisfied. Therefore the counts for these edges and its prefixes would be increased as Line 24 shows in the first 5 columns. In the 6-th column, start $= 1$, end $= 6$, $t_1 = 3$, $t_6 = 17$, $t_1 + 10 < t_6$ so DecrementCounts is performed on $e_{\text{start}} = e_1 = (a, b)$. As a result, counts[$(a, b)$] = counts[$(a, b)$] $-1 = 2$, counts[$(a, b)(c, b)$] = counts[$(a, b)(c, b)$] - counts[(a, b)] = $4 - 2 = 2$. The 7-th column

conducts DecrementCounts on $e_2 = (c, b)$ and the 8-th column performs IncrementCounts on $e_6 = (c, b)$.

As a result, there are in total 4 occurrences with center node b and neighbor pair (a, c). Thus the high-order proximity between a and b, a and c are both 4, as shown in Matrix (1). Note that the proximity between b and c is 5 instead of 4, because there is another occurrence of M, say $\langle (c, b, 17), (e, b, 25), (e, b, 27)\rangle$.

Algorithm 1 Temporal high-order proximity calculation.

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: An Ethereum transaction graph, $M_1, \ldots, M_s$: 3-node, ℓ-edge, δ-time temporal motifs.
Output: The temporal high-order proximity matrix $\mathcal{W}_{M_1}, \ldots, \mathcal{W}_{M_s}$.

Initialize the co-occurrence counts matrix $\mathcal{W}_{M_j}$ to be zero, $j = 1, \ldots, s$;
for $\alpha \in \mathcal{V}$ **do**
  Get all the neighbors of node v in $\mathcal{G}$ to be $\mathcal{N}_\alpha$;
  for (β, γ) *in all pairs of* $\mathcal{N}_\alpha$ **do**
    Gather all the temporal edges between α and β, α and γ;
    Sort these edges according to the timestamps and obtain an ordered sequence $S_\alpha = \{(u_i, v_i, t_i)\}_{i=1}^{L}$ with $t_1 < \ldots < t_L$;
    counts = MotifCounts(S_α);
  for $j = 1, \ldots, s$ **do**
    Increase the similarity between α and β, α and γ, β and γ in $\mathcal{W}_{M_j}$ by counts[M_j]
return $\mathcal{W}_{M_1}, \ldots, \mathcal{W}_{M_s}$.
Function *MotifCounts(S)*
  Initialize counts to be zero for every prefix and suffix of the frequent motifs;
  start = 1;
  for end = $1, \ldots, L$ **do**
    while $t_{\text{start}} + \delta < t_{\text{end}}$ **do**
      DecrementCounts(e_{start})
      start $+ = 1$
    IncrementCounts(e_{end})
  return counts;
Procedure *DecrementCounts(e)*
  counts[e] $- = 1$;
  for *suffix in counts.keys* ***and*** *suffix.length* $< \ell - 1$ **do**
    counts[concat(e,suffix)] $- =$ counts[suffix]
Procedure *IncrementCounts(e)*
  for *prefix in counts.keys.reverse()* ***and*** *prefix.length* $< \ell$ **do**
    counts[concat(prefix, e)] $+ =$ counts[prefix]
  counts[e] $+ = 1$;

If we take the transaction count as the compared low-order proximity, as shown in Matrix (1), the temporal high-order proximity matrix (left) differs from low-order proximity matrix (right) in two aspects. On one hand, some pairs of the nodes have non-zero

Table 1 An example execution of Algorithm 1

Start	1	1	1	1	1	1	2	3
End	1	2	3	4	5	6	6	6
t_{start}	3	3	3	3	3	3	5	7
t_{end}	3	5	7	8	11	17	17	17
Counts$[(a,b)]$	1	1	2	3	3	2	2	2
Counts$[(c,b)]$	0	1	1	1	2	2	1	2
Counts$[(a,b)(c,b)]$	0	1	1	1	4	2	2	4
Counts$[(c,b)(a,b)]$	0	0	1	2	2	2	0	0
Counts$[(a,b)(c,b)(c,b)]$	0	0	0	0	1	1	1	3
Counts$[(c,b)(a,b)(a,b)]$	0	0	0	1	1	1	1	1

high-order proximity but zero low-order proximity, such as a and c, d and f. This is because these pairs of nodes may co-occur in the same frequent transaction pattern but have no direct transaction between each other. On the other hand, the nodes under the high-order proximity tend to cluster into communities like $\{\{a, b, c\}, \{d, e, f\}\}$ while they are scattered under the low-order proximity. Therefore, we detect communities on Ethereum based on the high-order proximity.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 4 & 4 & 0 & 0 & 0 \\ 4 & 0 & 5 & 0 & 1 & 0 \\ 4 & 5 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 & 3 \\ 0 & 1 & 1 & 3 & 0 & 3 \\ 0 & 0 & 0 & 3 & 3 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 3 & 0 & 0 & 0 & 0 \\ 3 & 0 & 3 & 0 & 4 & 1 \\ 0 & 3 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix} \end{array} \tag{1}$$

The time complexity of Algorithm 1 is analysed as follows. The algorithm generally enumerates every node in the transaction graph for high-order proximity calculation. Hence, the outermost loop (Line 2) of Algorithm 1 takes $O(|\mathcal{V}|)$ time. For every node v, we need to iterate every node pair consisting of v and its direct neighbor thus the inner loop (Line 4) of Algorithm 1 takes $O(N_{\text{nbr}}^2)$, where N_{nbr} is the largest number of neighbors for the nodes in $\mathcal{V}$. The sorted sequence could be obtained in $O(1)$ time if we assume that the temporal edges between two nodes are logged in their order of occurrence. The complexity of Function *MotifCounts* (Line 11) depends on the updating number of counts. The number of keys in counts is proportional to 2^ℓ. Each temporal edge in the ordered sequence is processed twice as it enters and leaves the time window. Hence the update complexity is $O(2^\ell|S|)$ where $|S|$ is the largest length of the ordered sequence. Therefore, the overall complexity of Algorithm 1 is $O(|\mathcal{V}||S|N_{\text{nbr}}^2 2^\ell)$.

4.3 Temporal high-order aware community detection

In this subsection, we detail the third technical component, i.e. the temporal high-order aware community detection method, which incorporates the frequent temporal motif into the original transaction graph for better encoding high-order community structure and avoiding the hypergraph fragmentation issue [16].

The temporal motif based hypergraph $\mathcal{G}_M = (\mathcal{V}_M, \mathcal{E}_M)$ could be constructed from the high-order proximity matrix $\mathcal{W}_M$. Note that we only consider those nodes which have non-zero high-order proximity with at least one other node thus $\mathcal{V}_M \subset \mathcal{V}$. $\mathcal{E}_M$ denotes the edges corresponding to the non-zero elements in $\mathcal{W}_M$. Accordingly, a set of c connected components of the hypergraph $\mathcal{G}_M$ could be identified $\Phi = \{\phi_i\}_{i=1}^{c}$. Suppose that the components are sorted in a descending order according to their node size. We collect the edges in top-K_c components to be the enhanced edge set $\mathcal{E}_\Phi$.

Finally, the transaction graph could be enhanced as $\mathcal{G}_H = (\mathcal{V}_M, \mathcal{E}_{\mathcal{V}_M} \cup \mathcal{E}_\Phi)$ where $\mathcal{E}_{\mathcal{V}_M} \subset \mathcal{E}$ is the edges among the nodes in $\mathcal{V}_M$. The partition $\mathcal{C}$ of $\mathcal{V}_M$ could be obtained by optimizing the modularity Q of $\mathcal{G}_H$ and its general form is (2), where $m = m(\mathcal{G}_H)$ is the number of edges in $\mathcal{G}_H$, $m_{in} = m_{in}(\mathcal{C}, \mathcal{G}_H)$ is the number of intra-community edges in the partition $\mathcal{C}$ of $\mathcal{G}_H$, $m'_{in} = m'_{in}(\mathcal{C})$ is the number of intra-community edges in the random graphs constructed from $\mathcal{C}$.

$$Q(\mathcal{C}, \mathcal{G}_H, \gamma) = \frac{m_{in}}{m} - \gamma \frac{\mathbb{E}(m'_{in})}{m} \tag{2}$$

The basic idea is to consider the fraction of intra-community edges among all the edges, which corresponds to the first term of (2). But this can be easily optimized by a trivial partition which is exactly the graph itself. We penalize this trivial solution by a random graph model whose degree sequence is akin to that of graph $\mathcal{G}_H$. For this model, the probability that node i and j are connected equals $\frac{d_i d_j}{2m}$, where d_i is the degree of node i. Then we have $\mathbb{E}(m'_{in}) = \frac{1}{4m} \sum_{C \in \mathcal{C}} D(C)^2$, where $D(C) = \sum_{i \in \mathcal{C}} d_i$ and the standard form is shown in (3).

$$Q(\mathcal{C}, \mathcal{G}_H, \gamma) = \frac{m_{in}}{m} - \frac{\gamma}{4m^2} \sum_{C \in \mathcal{C}} D(C)^2 \tag{3}$$

At the beginning, each node forms its own community so that the number of communities equals the number of nodes. The first phase is iterated over all nodes: for each node, we evaluate the modularity gain by moving node i to the community of its neighbor, and then the node i is placed in the community with the largest modularity gain, as long as it is positive. If there is no possible positive gain, node i remains in the original community. The second phase is to merge the nodes in the same community to construct a supernode. The weights of links between supernode are given by the sum of the weight of the links between nodes in the original communities. Then the first phase is repeated with the supernodes, and the iteration process is keeping elapsed until the partition $\mathcal{C}$ stay unchanged or the maximum iteration threshold is accessed.

5 Experiment

In this section, we investigate the performance of THCD and conduct experiments to answer the following research questions:

- **RQ1**. What are the characteristics of the transaction patterns in the Ethereum transaction graph?
- **RQ2**. How does our method benefit the behavior analysis on Ethereum?
- **RQ3**. What are the suspicious activities like in the detected communities by our method?

Implementation details. All the experiments are conducted on a Ubuntu 16.04 server, with 40 cores Intel(R) Xeon(R) CPU E5-2640 v4 @2.40GHz and 128 GB memory. The

high-order proximity computation algorithm is implemented in C++ under the framework of SNAP [15] and the other components of THCD are implemented with Python 3.7.3.

5.1 Dataset

To interact with the Ethereum network, we build a Go Ethereum client (Geth) to synchronize the blocks on Ethereum. We choose 2 million blocks from Jan 30, 2018 to Jan 3, 2019 to construct the dataset for this paper. Those transaction blocks are split into 4 datasets, each containing 0.5 million blocks. The detailed statistics are given in Table 2.

To construct the transaction graph from the Ethereum blocks, we treat each account address as a node and each transaction as a directed temporal edge. However, most of the accounts have few transactions during the period of the blocks, and we thus remove those accounts that have less than 20 transaction records in these blocks. This is reasonable in our approach since these accounts are impossible to take part in any frequent transaction pattern due to their few occurrences. As a result, around $3\% \sim 4\%$ of the accounts in the blocks are included in the transaction graphs and the detailed statistics are summarized in Table 2.

5.2 Transaction pattern mining

After the transaction graph construction, transaction patterns are mined by calculating their support in the transaction graphs. In this paper, we choose to mine 3-node, 3-edge, 60 seconds-time temporal motifs from the transaction graph for the following reasons. Firstly, the 2-node motifs are improper to our problem since they cannot represent interactive relations among multiple addresses. Though 4-node or even larger motifs may contain more useful information, the complexity for mining them in large graphs will be very high, which may render our mining algorithm inefficient to discover the frequent transaction patterns. Hence, 3-node motifs are selected. Secondly, the 2-edge motifs only record one transaction between two address, which fails to encode high-order interactions among address. Hence, they are filtered out. More edges would significantly increase the computational burden in mining the frequent transaction patterns, and thus are discarded as well. Finally, the temporal threshold, namely 60 seconds for the temporal interval, was chosen by hyper-parameter tuning. It is related to the size of transaction graph in Table 2 and the suspicious behaviors of the accounts. If it is set to be larger, it is less likely that the related accounts are suspicious. If it is smaller, the nodes in the transaction graph would be fewer.

As mentioned in Section 4.1, the counting algorithm in [22] is employed to count the occurrences of all the 3-node, 3-edge, 60 seconds-time temporal motifs (as a total of 32 kinds of motifs) to decide which of them are frequent. To demonstrate the characteristic of the transaction patterns in the Ethereum transaction graph, we exhibit the supports of top-16 motifs in Figure 4. The motifs are arranged according to the descending order of their

Table 2 Statistics of datasets

Name	Start block	End block	#Account	#Trans	#Node	#Edge
$\mathcal{G}_A$	5,000,000	5,500,000	9,677,940	34,329,363	320,299	12,865,400
$\mathcal{G}_B$	5,500,000	6,000,000	8,966,203	32,388,404	319,572	13,612,208
$\mathcal{G}_C$	6,000,000	6,500,000	6,653,841	25,579,269	248,103	11,438,459
$\mathcal{G}_D$	6,500,000	7,000,000	5,780,602	22,691,388	233,864	10,585,189

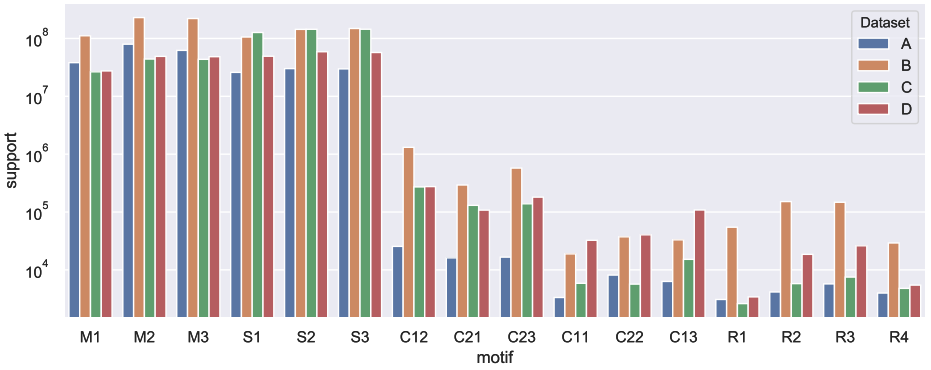

Figure 4 The support of top-16 motifs in four transaction graphs. The horizontal axis denotes the name of the motifs and the vertical axis is the support of the motifs. Different colors indicate different datasets

support, and the supports of top-6 temporal motifs exceed 10 million. These temporal motifs are significantly more frequent than others. The next following three temporal motifs (c.f. C12, C21, C23 in Figure 4), although their supports are lower than the top-6, they occur more frequently than the remaining ones for all the four datasets. Hence they are deemed to be frequent as well. As a result, we altogether adopt a total of 9 temporal motifs as the transaction patterns for the following evaluations.

These frequent temporal motifs reflect some characteristics of the common trading patterns in the Ethereum network. For the top-3 temporal motifs, their temporal edges end with the same node, and they differ in the time order of the three temporal edges. Since they reflect that two accounts transfer fund to the same account, they are named MERGE transaction patterns, as shown in Figure 5a. The notations for SPLIT transaction patterns (S1 $\sim$ S3) are akin to the MERGE patterns except the directions of transactions. For the SPLIT patterns, their temporal edges start from the same node, and they may be related to fund split transactions. The third class in the frequent temporal motif is named CHAIN. For the completeness, we list all the six variants of the CHAIN patterns in Figure 5b, but only three of them are mined to be frequent in our experiments. From the counting results in Figure 4, the supports of temporal motif C12, C21, C23 are apparently higher than the other three, so we

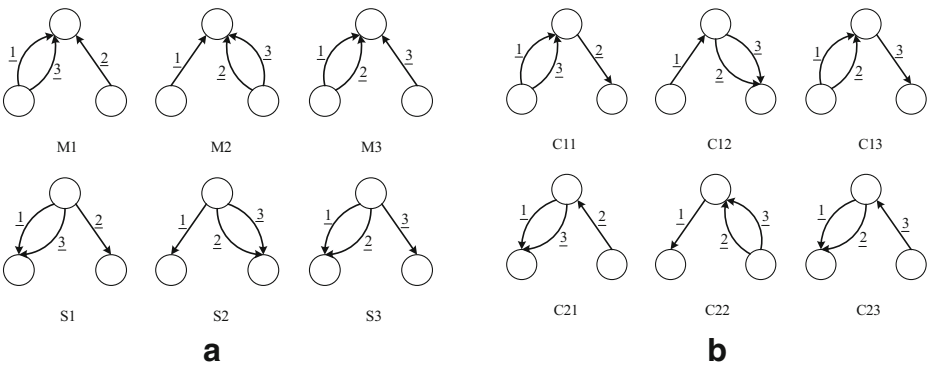

Figure 5 The discovered transaction patterns. The all patterns in (**a**) and C12, C21, C23 in (**b**) are frequent in our datasets. The numbers with underline indicate the time order

deem these three temporal motifs as transaction patterns. We thus derive nine transaction patterns in our datasets.

5.3 Detect $\mathcal{W}_M$-based account community

Next, we leverage these transaction patterns to discover the account communities based on the temporal high-order proximities among accounts and understand the behaviors of the accounts on Ethereum. Since there are few methods for Ethereum behavior analysis that can be directly compared with the proposed THCD, we focus on comparing our method with state-of-the-art community detection methods.

Baseline Methods. We compare THCD with classic community detection method **Louvain** [3] and two state-of-the-art high-order community detection methods **Motif** [2] and **EdMot** [16]. For each frequent temporal motif, Louvain performs partition on the original transaction graph and Motif takes the high-order proximity as input. EdMot exploits both the original transaction graph and its edge enhancement strategy.

Evaluation Metric. We adopt **Modularity** to evaluate the quality of the discovered community structure. A higher value of modularity indicates better community partition. K_c is set to be 50 for EdMot and THCD.

Results. Tables 3, 4 and 5 demonstrate the Modularity of the compared methods over each transaction patterns on the four datasets. From these tables, we observe that our THCD can achieve the best performance on most of the test cases. The mean of Modularity of THCD in each class of transaction pattern (c.f. the rightmost column in every table) outperforms the baselines in 10 cases out of the total 12 comparisons. For baselines, Louvain detects communities only based on low-order proximity and do not explore the temporal information thus performs worse than high-order methods like Edmot and THCD. We also observe that the high-order method Motif performs poorly in our datasets, and we

Table 3 Modularity of community detected in transaction graphs for MERGE motif

Graph	Method	M1	M2	M3	Avg
$\mathcal{G}_A$	Louvain	0.8020	0.8094	0.6578	0.7564
	Motif	0.7084	0.7157	0.7261	0.7167
	EdMot	0.8127	**0.8141**	0.8132	0.8133
	THCD	**0.8130**	0.8138	**0.8136**	**0.8135**
$\mathcal{G}_B$	Louvain	0.8193	0.8200	0.8205	0.8199
	Motif	0.6414	0.6366	0.6624	0.6468
	EdMot	**0.8244**	0.8189	0.8221	0.8218
	THCD	**0.8244**	**0.8214**	**0.8233**	**0.8230**
$\mathcal{G}_C$	Louvain	0.8413	0.8447	0.8452	0.8437
	Motif	0.7836	0.7482	0.7517	0.7612
	EdMot	0.8419	0.8462	0.8459	0.8447
	THCD	**0.8431**	**0.8472**	**0.8468**	**0.8457**
$\mathcal{G}_D$	Louvain	0.8508	0.8558	0.8535	0.8534
	Motif	0.7433	0.7675	0.7508	0.7539
	EdMot	0.8528	0.8571	**0.8552**	0.8550
	THCD	**0.8530**	**0.8575**	0.8547	**0.8551**

Table 4 Modularity of community detected in transaction graphs for SPLIT motifs

Graph	Method	S1	S2	S3	Avg
$\mathcal{G}_A$	Louvain	0.7813	0.7722	0.7719	0.7751
	Motif	0.6660	0.6546	0.6724	0.6643
	EdMot	0.8066	0.7979	0.7997	0.8014
	THCD	**0.8071**	**0.7997**	**0.8000**	**0.8023**
$\mathcal{G}_B$	Louvain	0.7938	0.7762	0.7901	0.7867
	Motif	0.6141	0.6123	0.6038	0.6101
	EdMot	**0.7947**	**0.7765**	**0.7904**	**0.7872**
	THCD	0.7942	0.7764	**0.7904**	0.7870
$\mathcal{G}_C$	Louvain	0.6561	0.6229	0.6194	0.6328
	Motif	0.5136	0.4294	0.4514	0.4648
	EdMot	0.6564	0.6224	0.6193	0.6327
	THCD	**0.6568**	**0.6240**	**0.6200**	**0.6336**
$\mathcal{G}_D$	Louvain	0.8676	0.8522	0.8610	0.8603
	Motif	0.7284	0.6782	0.7096	0.7054
	EdMot	0.8681	**0.8537**	0.8609	0.8609
	THCD	**0.8684**	**0.8537**	**0.8621**	**0.8614**

conjecture the reason could be that it suffers from the hypergraph fragmentation issue. For EdMot, it achieves competitive performance due to its enhancement in incorporating edge information into the high-order proximity calculation. To sum up, benefiting from the frequent temporal motifs, our method buoys a better community partition with a larger Modularity derived by the temporal high-order proximity.

Table 5 Modularity of community detected in transaction graphs for CHAIN motifs

Graph	Method	C12	C21	C23	Avg
$\mathcal{G}_A$	Louvain	0.6131	0.5747	0.5474	0.5784
	Motif	0.5292	0.5762	0.5866	0.5640
	EdMot	0.7353	0.7270	**0.7368**	0.7330
	THCD	**0.7370**	**0.7314**	0.7344	**0.7343**
$\mathcal{G}_B$	Louvain	0.6067	0.5911	0.6239	0.6072
	Motif	0.4895	0.5938	0.6581	0.5805
	EdMot	**0.7398**	**0.7912**	0.8159	**0.7823**
	THCD	0.7377	0.7860	**0.8163**	0.7800
$\mathcal{G}_C$	Louvain	0.7239	0.6394	0.7129	0.6921
	Motif	0.6500	0.7064	0.7398	0.6987
	EdMot	**0.8473**	0.8473	0.8422	0.8456
	THCD	0.8471	**0.8491**	**0.8522**	**0.8495**
$\mathcal{G}_D$	Louvain	0.8237	0.5987	0.6361	0.6862
	Motif	0.6678	0.7056	0.7522	0.7085
	EdMot	0.8310	0.8314	0.8251	0.8292
	THCD	**0.8314**	**0.8326**	**0.8260**	**0.8300**

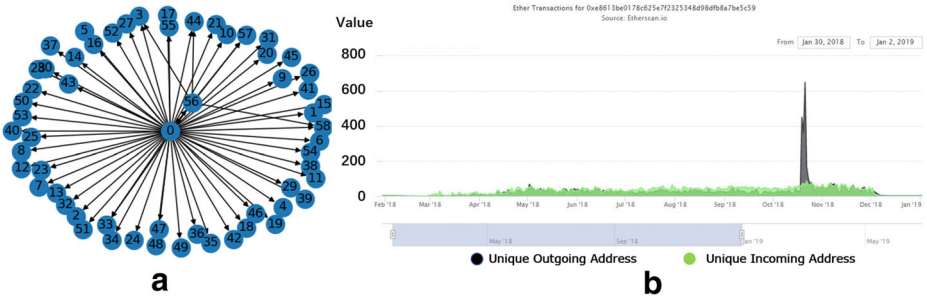

Figure 6 One of the detected community in $\mathcal{G}_D$ under the transaction pattern S1 and the analytics of the suspicious account "0" from etherscan.io. The accounts in this community are indexed by numbers

5.4 Case study of Ethereum behavior understanding

Next, we look closer to the detected communities to demonstrate what THCD could find by some specific interesting accounts.

Figure 6a shows a community of 59 nodes and 1247 temporal edges in the dataset $\mathcal{G}_D$, which is constructed from Oct 12, 2018 to Jan 3, 2019. The high-order proximity for this community is defined by S1, which is a SPLIT transaction pattern. We omit the timestamps in each temporal edge for the sake of brevity and the total occurrences of S1 in this community is 96. We find there is a center node denoted by number 0 in such community which transferred fund to other 58 nodes consistently in a short period of time. And we analyze the transaction history of the node 0 from Jan 30, 2018 to Jan 2, 2019 in Figure 6b. From the figure, we can see the number of unique incoming address remains almost the same in this year while the number of unique outgoing address increases suddenly at the end of October, 2018, which corresponds to the anomaly found by our method. It would be hard for those statistical methods to detect such anomalous features since they usually extract account features from a long period of time. The low-order detection methods could also hardly find this kind of community since low-order proximity does not consider high-order interactions among addresses. We emphasize that it is the temporal high-order proximity that helps our approach to uncover such suspicious accounts group.

Figure 7 demonstrates part of the communities detected in $\mathcal{G}_D$, and the high-order proximity is defined based on C23, a CHAIN transaction pattern. We visualize part of the communities with the known addresses owned by some exchanges labeled in the figure. The considered CHAIN pattern, i.e. C23, occurs very frequently in these communities. We conjecture that the CHAIN transaction patterns across different exchanges may be the practice of arbitrage, where people take advantage of the price difference to earn profits. These illicit trading transactions are expected to finish within a short time to evade significant price fluctuation, and thus these kinds of behaviors could be captured under the mined frequent 3-node, 3-edge, 60 seconds-time transaction patterns. Besides, the CHAIN patterns within the same exchange may correlate to exaggerated trade volumes of the exchange accounts. Ethers out of the withdrawal addresses of exchange could be transferred back to its deposit addresses, which makes an illusion that the exchange is popular on Ethereum without loss of its balance. Our observation coincides with authoritative report[4] published by Crypto

[4] https://medium.com/crypto-integrity/fake-volumes-in-cryptocurrency-markets-february-report-fec9329f1f98

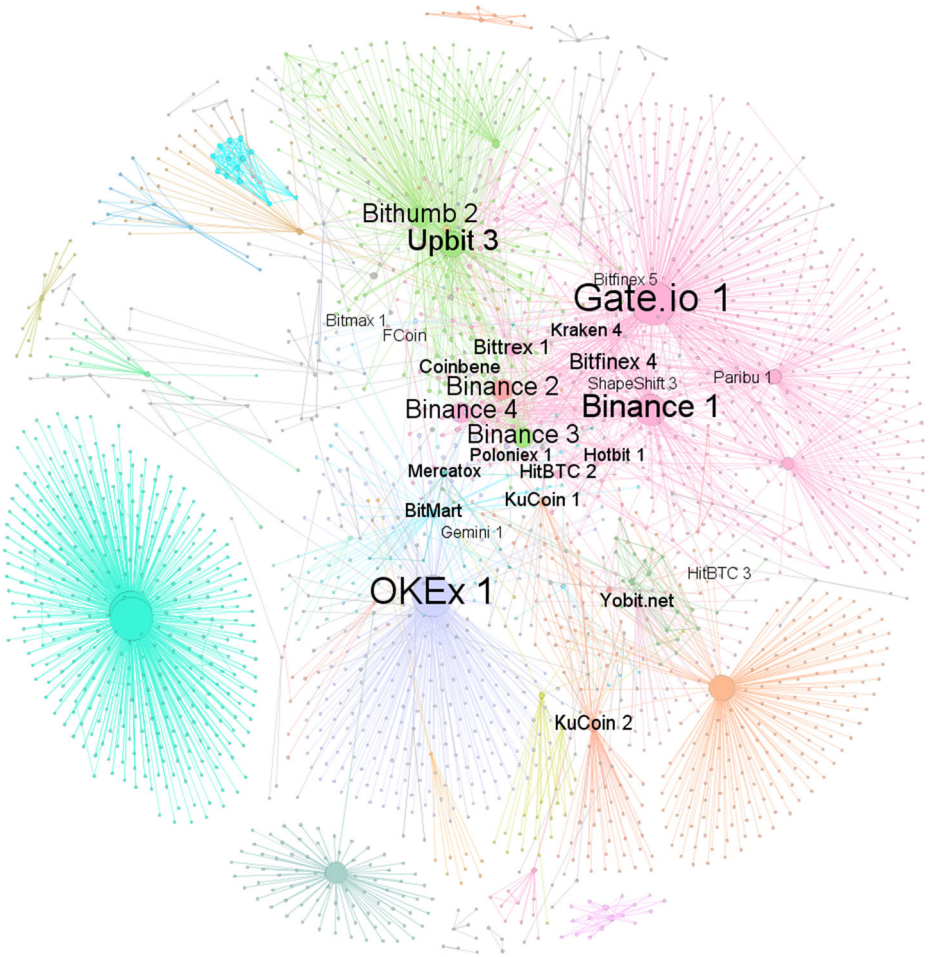

Figure 7 Visualization of detected communities in $\mathcal{G}_D$ under the transaction pattern C23. The known addresses owned by exchanges are annotated on the nodes. The nodes in the same color indicate they belong to the same community. The labeled node "OKEx 1" belongs to the community in the color of purple

Integrity, and they have identified malicious fake volume practice in top exchanges including OKEx, which is exactly confirmed by our method (see the labeled node "OKEx 1" near the center of Figure 7). It is an encouraging result that our approach might be able to reveal plausible evidence for validating abnormal accounts.

5.5 Scalability

Finally, we verify the efficiency of our approach. Recall that the most time-consuming part of THCD is the high-order proximity computation algorithm, i.e., $O(|\mathcal{V}||S|N_{\text{nbr}}^2 2^{\ell})$ (cf. Section 4.2). Thus we mainly evaluate the scalability of Algorithm 1 as we expand the number of blocks or increase the computing threads. We adopt a straightforward parallel strategy that assigns the nodes to different threads when computing their occurrence in different temporal motifs.

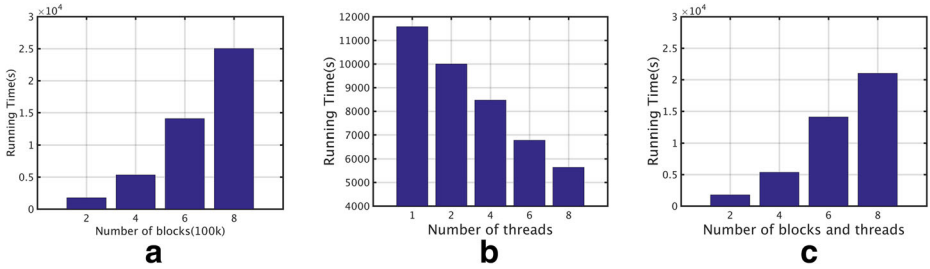

Figure 8 Scalability of THCD

The basic number of blocks is 0.2 million in this evaluation. First, we investigate the performance of THCD by increasing the computing blocks from 0.2 million to 0.8 million and the total running time is shown in Figure 8a. Since the complexity with respect to the largest number of neighbors N_{nbr} is square, it is hard for THCD to achieve a linear sizeup but the running time is adoptable in practice.

To study the strong scalability of THCD, we keep the number of blocks to be 0.5 million which is the same scale as the transaction graph in the experiments and vary the computing thread from 1 to 8. From the results in Figure 8b, we observe that THCD is parallelizable using multi-thread counting algorithm. The acceleration is reduced when the thread is set to be 8 because a few nodes have a large degree and the neighbor-based node pair enumeration of the algorithm dominates the main complexity, which has not been paralleled in our strategy.

Finally, to test the weak scalability, we keep the threads and blocks growing in pace with each other and the performance is shown as Figure 8c. The running time increases with the simultaneous increase of both threads and blocks. For better scaleup performance, the parallel strategy needs to be further improved, such as partitioning the complex operation of nodes with large degree into different threads, which will be left as future work. Nevertheless, from the current results, we could conclude THCD is scalable to large-scale transaction graphs.

6 Conclusion

In this paper, we investigated the problem of behavior analysis on Ethereum with advanced techniques in graph analysis. We proposed THCD, which is an account community detection approach incorporating both temporal information of Ethereum transactions and high-order interactions among accounts. It centers on three technical components. First, to consider temporal interactions among multiple accounts in the platform, frequent temporal motifs consisting of three nodes and three temporal edges are mined as transaction patterns. Second, the temporal high-order proximity is computed under the guidance of these discovered patterns. Finally, a temporal high-order aware community detection method is devised to partition the corresponding accounts into sub-groups. Experiments on two million Ethereum blocks demonstrate the effectiveness and efficiency of our method. The proposed THCD could not only detect potential suspicious account community but also reveal plausible evidences for confirmed abnormal accounts. Meanwhile, it is scalable to large-scale transaction graphs.

Acknowledgements The research work is supported by the National Key Research and Development Program of China under Grant No.2017YFB1002104, the National Natural Science Foundation of China under Grant No. 92046003, 61976204, U1811461, 61672499, Key Special Project of Beijing Municipal Science & Technology Commission (Z181100003218018). Xiang Ao is also supported by the Project of Youth Innovation Promotion Association CAS and Beijing Nova Program Z201100006820062. We thank Haibo Zhou for his valuable suggestions for this work.

References

1. Abdelhamid, E., Canim, M., Sadoghi, M., Bhattacharjee, B., Chang, Y.C., Kalnis, P.: Incremental frequent subgraph mining on large evolving graphs. IEEE Transactions on Knowledge and Data Engineering (2017)
2. Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. Science (2016)
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment (2008)
4. Bogner, A.: Seeing is Understanding: Anomaly Detection in Blockchains with Visualized Features. In: Ubicomp (2017)
5. Chen, T., Zhu, Y., Li, Z., Chen, J., Li, X., Luo, X., Lin, X., Zhange, X.: Understanding Ethereum via graph analysis. In: INFOCOM (2018)
6. Chen, W., Zheng, Z., Cui, J., Ngai, E., Zheng, P., Zhou, Y.: Detecting ponzi schemes on Ethereum: Towards healthier blockchain technology. In: WWW (2018)
7. Chen, W., Zheng, Z., Ngai, E.C., Zheng, P., Zhou, Y.: Exploiting blockchain data to detect smart ponzi schemes on ethereum. IEEE Access **7**, 37575–37586 (2019)
8. Farrugia, S., Ellul, J., Azzopardi, G.: Detection of illicit accounts over the ethereum blockchain. Expert Syst. Appl. **150**, 113318 (2020)
9. Guimera, R., Amaral, L.A.N.: Functional cartography of complex metabolic networks. Nature (2005)
10. Gurukar, S., Ranu, S., Ravindran, B.: Commit: a scalable approach to mining communication motifs from dynamic networks. In: SIGMOD (2015)
11. Jian, X., Lian, X., Chen, L.: On efficiently detecting overlapping communities over distributed dynamic graphs. In: ICDE (2018)
12. Jiang, Y., Huang, X., Cheng, H., Yu, J.X.: Vizcs: Online searching and visualizing communities in dynamic graphs. In: ICDE (2018)
13. Kleinberg, B., Kamps, J.: To the moon: defining and detecting cryptocurrency pump-and-dumps. Crime Science (2018)
14. KUMARI, R., CATHERINE, M.: Anomaly detection in blockchain using clustering protocol. International Journal of Pure and Applied Mathematics (2018)
15. Leskovec, J., Sosič, R.: Snap: A general-purpose network analysis and graph-mining library ACM Transactions on Intelligent Systems and Technology (TIST) (2016)
16. Li, P.Z., Huang, L., Wang, C.D., Lai, J.H.: Edmot: an edge enhancement approach for motif-aware community detection. In: KDD (2019)
17. Li, T., Shin, D., Wang, B.: Cryptocurrency pump-and-dump schemes. Available at SSRN 3267041 (2019)
18. Lin, D., Wu, J., Yuan, Q., Zheng, Z.: Modeling and understanding ethereum transaction records via a complex network approach. IEEE Transactions on Circuits and Systems II:, Express Briefs, pp. 1–1 (2020)
19. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science (2002)
20. Newman, M.E.: Finding community structure in networks using the eigenvectors of matrices. Physical review E (2006)
21. O'Kane, E.: Detecting patterns in the ethereum transactional data using unsupervised learning. Master's thesis, University of Dublin Trinity College (2018)
22. Paranjape, A., Benson, A.R., Leskovec, J.: Motifs in temporal networks. In: WSDM (2017)
23. SAYADI, S., REJEB, S.B., CHOUKAIR, Z.: Anomaly detection model over blockchain electronic transactions. In: IWCMC (2019)
24. SINGH, A.: Anomaly Detection in the Ethereum Network. Master's thesis, Indian Institute of Technology Kanpur (2019)
25. Sun, H., Ruan, N., Liu, H.: Ethereum analysis via node clustering. In: ICNSS (2019)

26. Tsourakakis, C.E., Pachocki, J., Mitzenmacher, M.: Scalable motif-aware graph clustering. In: WWW (2017)
27. Weber, M., Domeniconi, G., Chen, J., Weidele, D.K.I., Bellei, C., Robinson, T., Leiserson, C.E.: Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. In: Workshop of KDD (2019)
28. Wei, X., Lu, C., Ozcan, F.R., Chen, T., Wang, B., Wu, D., Tang, Q.: A behavior-aware profiling of smart contracts. In: International Conference on Security and Privacy in Communication Systems, pp. 245–258. Springer (2019)
29. Xu, J., Livshits, B.: The anatomy of a cryptocurrency pump-and-dump scheme. In: {USENIX} Security Symposium (2019)

Affiliations

Xiang Ao[1,2] · Yang Liu[1,2] · Zidi Qin[1,2] · Yi Sun[2,3] · Qing He[1,2]

Yang Liu
liuyang17z@ict.ac.cn

Zidi Qin
qinzidi20s@ict.ac.cn

Yi Sun
sunyi@ict.ac.cn

Qing He
heqing@ict.ac.cn

1 Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, China

2 University of Chinese Academy of Sciences, Beijing, China

3 Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China